# Exercise Sheet 3

1. How can matrix factorizations help you to compute determinants? Find $\det A$ from the following factorizations:

   (a) LU factorization $A = LU$ ($U$ upper triangular, $L$ lower triangular with 1's in the diagonal)

   (b) LU factorization (with partial pivoting) $PA = LU$ ($P$ permutation matrix)

   (c) Cholesky factorization $A = R^*R$ ($R$ upper triangular),

   (d) QR factorization $A = QR$ ($Q$ orthogonal/unitary, $R$ upper triangular).

   Consult the online Matlab documentation to find out how determinants are computed in Matlab.

2. Write a program that, given a regular $n \times n$ matrix $A$, computes the $LU$ factorization using Gaussian elimination without pivoting, if possible. Optimize your code by implementing he following features.

   (a) Avoid explicit forming of $L$ and $U$. Instead, store the coefficients $\ell_{ij}$ and $u_{ij}$ directly into the input matrix $A$. (After the last elimination step, the upper triangular part of $A$ should be equal to $U$, while subdiagonal elements of $A$ should be the subdiagonal elements of $L$.)

   (b) Use only one `for`-loop. The inner `for`-loop (i.e. the one which iterates through the subdiagonal rows) can be avoided by using an outer product formulation of the elimination step.

   Upload your program via Moodle.

3. Show that, if unnecessary zero operations are avoided, the number of flops required for computing an $LU$ factorization of a *band matrix* $A$ via Gaussian elimination is linear in $n$.

4. Compute the Cholesky factorization of

$$\begin{bmatrix} 2 & 1 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ 0 & 1 & 2 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix}.$$

5. In the lecture we have seen that every SPD matrix admits a Cholesky factorization.

Show that the *converse* statement is also true: If $A \in \mathbb{K}^{n\times n}$ has a Cholesky factorization, that is, $A = R^*R$ for some upper triangular matrix $R$ with positive diagonal entries, then $A$ is SPD.

6. Numerically testing whether a given matrix is SPD is not straightforward. Self-adjointness is easy to check, but positive definiteness is more challenging, since computation of determinants, for instance, cannot be done very efficiently. Exercise 5 shows that the following two statements are *equivalent:*

   - $A$ is SPD.
   - $A$ has a Cholesky factorization.

   This means that an algorithm implementing Cholesky factorization must fail, if $A$ is not positive definite. Consequently, Cholesky factorization can be used as a test for positive definiteness, which in general is much more efficient than computing determinants or eigenvalues.

   (b) How does Cholesky factorization fail if $A$ is not self-adjoint? How does it fail if $A$ is self-adjoint but not positive definite? (A rigorous proof is not required here. You can examine examples of such matrices.)

   (c) Write a program that tests for self-adjointness first, and then implements Cholesky factorization and terminates (with appropriate error messages), if $A$ is not SPD. Upload your program via Moodle.